A Convex Hull Algorithm using Point Elimination Technique

Dr. Sajjad Waheed, Tahmina Shirin, Md. Newaz Sharif, Md. Habibur Rahaman

Abstract— Graham's scan is an algorithm for computing the convex hull of a finite set of points in the 2D plane with time complexity O(nlogn). In the algorithm, points in 2D plane are sorted angle-wise and then selected as convex hull points by checking whether the points constitute right-turn or left-turn. A preprocessing technique on Graham's Scan algorithm has developed in this paper. First, the number of points was eliminated those are not exist on the hull. To do this, a quadrilateral was created using four optimal points and then create four triangles using another four optimal points. Thus the complexity of the algorithm O(nlogn) is expected to work much faster on smaller number of points.

Index Terms— - Convex Hull, Point elimination, Graham's scan algorithm, Point in 2D plane.

1 Introduction

AN object is convex in Euclidean space, if for every pair of points within the object, every point on the straight line segment that joins the pair of points is also within the object. Let S be a vector space over the real numbers, or, more generally, some ordered field. This includes Euclidean spaces. A set C in S is said to be convex if every point on the line segment connecting x and y is in C.

A convex polygon is a simple polygon whose interior is a convex set. The properties of a simple polygon are all equivalent to convexity, and can be stated as (i) Every internal angle is less than or equal to 180 degrees, and (ii) Every line segment between two vertices remains inside or on the boundary of the polygon.

A simple polygon is strictly convex if every internal angle is strictly less than 180 degrees. Equivalently, a polygon is strictly convex if every line segment between two non-adjacent vertices of the polygon is strictly interior to the polygon except at its endpoints.

In mathematics, the convex hull or convex envelope of a set X of points in the Euclidean plane or Euclidean space is the smallest convex set that contains X. For instance, when X is a bounded subset of the plane, the convex hull may be visualized as the shape formed by a rubber band stretched around X.

Formally, the convex hull may be defined as the intersection of all convex sets containing X or as the set of all convex combinations of points in X. With the later definition, convex hulls may be extended from Euclidean spaces to arbitrary real vector spaces; they may also be generalized further, to oriented matroids.

2 Literature Servey

Per-Olof Fjällström, Jyrki Katajainen, Christos Levcopoulos, Ola Petersson [27] presented a parallel algorithm for finding the convex hull of a sorted set of points in the plane.Their algorithm runs in O(logn/log logn) time using O(nlog logn/logn) processors in the Common crcw pram computational model, which is shown to be time and cost optimal. The algorithm is based on n 1/3 divide-and-conquer and uses a simple pointer-based data structure.

Kasun Ranga Wijeweera [28] proposed a new efficient algorithm to construct the convex hull of a set of points in the plane. The proposed algorithm is able to find the points on the convex hull in boundary traversal order. When the convex hull has collinear points, the algorithm can detect all the collinear points on the hull without skipping the intermediate points. Furthermore it can deal with the data sets were coincident points appear. Two main methods have been used to make the algorithm efficient. First one is achieving parallelism which is done by partitioning the data set. Second one is data reduction which is done by removing unnecessary points at each step of processing.

Mart M.McQUEEN and Godfried T. TOUSSIANT proposed e modification of Kirkpatrick and Seidel's algorithm. Kirkpatrick and Seidel's algorithm is capable of computing convex hull of n points in O(nlogh) worst case time, where h denotes the number of points on the convex hull of the set. But *M.McQUEEN and Godfried T. TOUSSIANT*'s [03] algorithm is believed to run in O(n) expected time for many reasonable distributions of points.

Herv´e Br¨onnimann, John Iacono, Jyrki Katajainen, Pat Morin, Jason Morrison, Godfried Toussaint proposed a space-efficient algorithm in which the output is given in the same location as the input and only a small amount of additional memory is used by the algorithm.They described four space-efficient algorithms for computing the convex hull of a planar point set.

Gang Mei, John C.Tipper and Nengxiong Xu [29] represent an alternate choice to compute the convex hull for planar point sets. At first they discard the interior points and then sort the remaining vertices by x and y coordinates separately. Then create a group of quadrilaterals recursively according to the sequence of sorted lists of points. Finally, the desired convex hull is built based on a simple polygon derived from all quadrilaterals.

1. C. Bradford Barber, David P. Dobkin, Hannu Huhdanpaa's method:

This algorithm is a practical convex hull algorithm that combines the two-dimensional Quickhull Algorithm with the general-dimension Beneath-Beyond Algorithm. It is similar to the randomized, incremental algorithms for convex hull and Delaunay triangulation. They provide empirical evidence that the algorithm runs faster when the input contains non extreme points and that it uses less memory. Computational geometry algorithms have traditionally assumed that input sets are well behaved. When an algorithm is implemented with floating-point arithmetic, this assumption can lead to serious errors. They briefly describe a solution to this problem when computing the convex hull in two, three, or four dimensions. The output is a set of "thick" facets that contain all possible exact convex hulls of the input. A variation is effective in five or more dimensions.

2. Per-Olof Fjällström, Jyrki Katajainen, Christos Levcopoulos, Ola Petersson's method:
They presented a parallel algorithm for finding the convex hull of a sorted set of points in the plane.Their algorithm runs in O(logn/log logn) time using O(nlog logn/logn) processors in theCommon crcw pram computational model, which is shown to be time and cost optimal. The algorithm is based on n 1/3 divide-and-conquer and uses a simple pointer-based data structure.

3. Kasun Ranga Wijeweera's method:
Kasun Ranga Wijeweera proposed a new efficient algorithm to construct the convex hull of a set of points in the plane. The proposed algorithm is able to find the points on the convex hull in boundary traversal order. When the convex hull has collinear points, the algorithm can detect all the collinear points on the hull without skipping the intermediate points. Furthermore it can deal with the data sets where coincident points appear. Two main methods have been used to make the algorithm efficient. First one is achieving parallelism which is done by partitioning the data set. Second one is data reduction which is done by removing unnecessary points at each step of processing.

4. Mart M.McQUEEN and Godfried T. TOUSSIANT's method:
Mart M.McQUEEN and Godfried T. TOUSSIANT proposed a modification of Kirkpatrick and Seidel's algorithm. Kirkpatrick and Seidel's algorithm is capable of computing convex hull of n points in O(nlogh) worst case time, where h denotes the number of points on the convex hull of the set .But M.McQUEEN and Godfried T. TOUSSIANT's algorithm is believed to run in O(n) expected time for many reasonable distributions of points.

5. Herv´e Br¨onnimann, John Iacono, Jyrki Katajainen, Pat Morin, Jason Morrison, Godfried Toussaint's method:

Herv´e Br¨onnimann, John Iacono, Jyrki Katajainen, Pat Morin, Jason Morrison, Godfried Toussaint proposed a space-efficient algorithm in which the output is given in the same location as the input and only a small amount of additional memory is used by the algorithm.They described four space-efficient algorithms for computing the convex hull of a planarpoint set.

6. Gang Mei, John C.Tipper and Nengxiong Xu's method:
Gang Mei, John C.Tipper and Nengxiong Xu represent an alternate choice to compute the convex hull for planar point sets. At first they discard the interior points and then sort the remaining vertices by x and y coordinates separately. Then create a group of quadrilaterals recursively according to the sequence of sorted lists of points. Finally, the desired convex hull is built based on a simple polygon derived from all quadrilaterals.

3 Proposed Method

To determine the convex hull of a given problem set, at first to minimize the points from which the convex hull is found. A huge number of points have removed and then the process will be faster. A large number of points was removed from the given problem set, which are not definitely hull points. The idea is to find 4 optimal points by forming a quadrilateral, eliminate the points that lie inside quadrilateral. After that find another 4 optimal points by triangulation and do the same.

3.1 Making a Quadrilateral

At first step find four boundary points, such as:
1. Vertically lowest (bottom-most) point
2. Vertically highest (top-most) point
3. Leftmost point
4. Rightmost point

Then using these four points draw a quadrilateral. To become a convex hull, the nodes inside the quadrilateral boundary can't be hull boundary points. So now avoid these points and remove from the data set to make the algorithm faster.

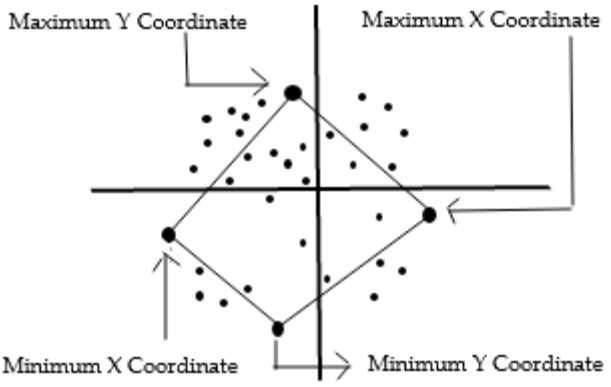


Fig. 1. Forming a quadrilateral.

To eliminate points in quadrilateral first add the point to the corner of a quadrilateral. In the next step, the areas of triangle are calculated. If the total summation of areas are not equal to the quadrilateral area then the point does not belong to the quadrilateral. Such a quadrilateral is shown in the figure 13. For the calculation, the following formulas are used:
Step 1: Circumference of the triangle calculation, S = a + b + c ; where a,b,c the sides of triangle.
Step 2: Area of the triangle = sqrt(S (S - a) (S - b) (S - c)).

There will be 4 triangles, Assume that triangle areas are area1, area2, area3, area4 and area of a quadrilateral is R.
Step 3: Determination of position of the point: If the area of quadrilateral R is not equal to the sum of area1, area2, area3 and area4, then the point is not inside the quadrilateral.

3.2 Triangulation

Triangulation is another optimization method that can be applied after the previous methods were used. From the previous quadrilateral finding process, found four edges were found, within which the desired point could be found. Connecting these edges, four triangles can be drawn. Let us consider the top-most and right-most points are two end points in the quadrilateral. There is a point that is laying at the furthest part from this line. This point is in the top-right side of the connecting line of two existing points is the third end point, which can be connected to form a triangle, as shown in the figure 3.

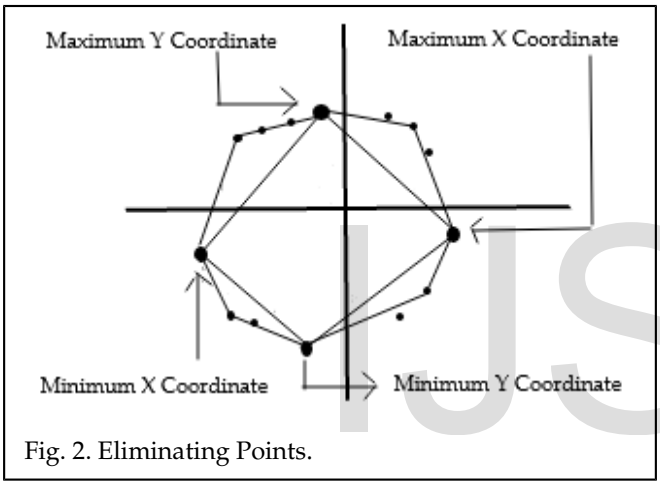
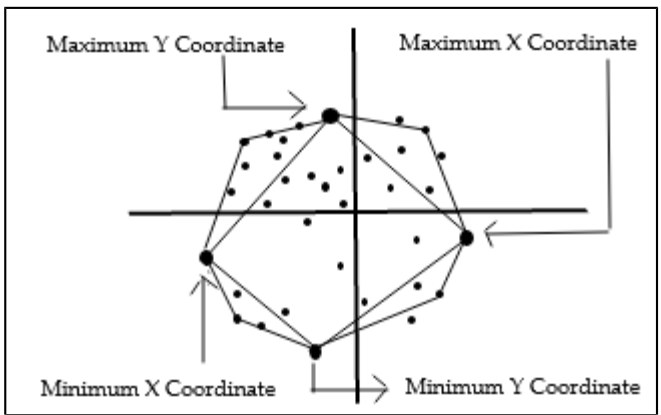


Fig. 2. Eliminating Points.



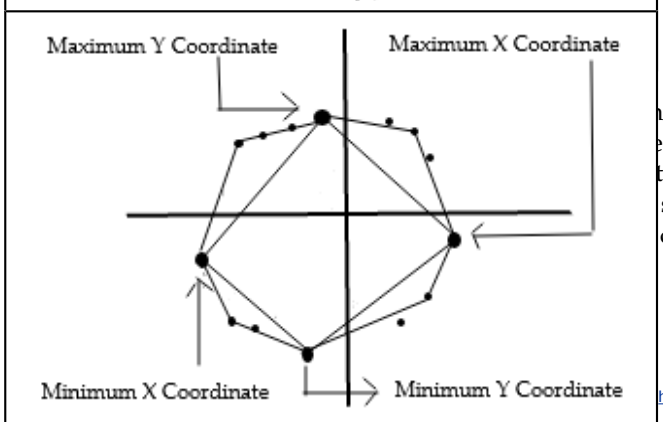


Fig. 4. Elimination of points within triangle.

At the end, eight boundary points and some points on boundary or outside the quadrilateral and triangles. Now apply any one of the O(nlogn) algorithms for convex hull on rest of the points which should be very few in number heuristically.

4 Result Obtained

Here the result shows, higher input size it shows better performance. It is because the numbers are generated evenly. So there are large number of points are eliminated. If the majority of the points are lying on the quadrilateral or the triangles, this modification will reduce the complexity of any O(nlogn). Specially for number of points greater than 10^5 it will show better performance.
If Although this solution shows better performance in larger input but it will show worse performance in input size of 100 or if the majority of points lie on the boundary of a polygon due to extra checking for point elimination.

5 Conclusion

Traditional Graham's Scan Algorithm is good enough for input size about 100 or 1000 but the concern is huge input like 10^5. In this proposed method, optimization totally depends on input pattern. If the points are scattered and dense it will show best performance with time complexity close to O(n) heuristically.

TABLE 1
Result Obtained And Comparison with Existing Method

Input Size	Point Elimination and Graham Scan's Algorithm	Only Graham Scan's Algorithm
100	0.00000 Sec	0.00000 Sec
10000	0.03100 Sec	0.04600 Sec
100000	0.04220 Sec	0.06700 Sec
1000000	4.69600 Sec	8.22100 Sec

References

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L.Rivest, Clifford Stein 3rd Edition (2010-2011). INTRODUCTION TO ALGORITHM.

December 12, 2012.
[2] Joseph O'Rourke 2ND Editon. COMPUTATIONAL GEOMETRY IN C. December 17, 2012.
[3] Mary M. McQUEEN and Godfried T. TOUSSAINT, On the ultimate convex hull algorithm in practice. January 01, 2013.
[4] Graham, R.L. (1972). An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set. Information Processing Letters 1, 132-133. January 12, 2013.
[5] McConnell, Jeffrey J. (2006), Computer Graphics: Theory Into Practice, p. 130, ISBN 0-7637-2250-2, December 22, 2012.
[6] A. Jarvis, On the identification of the convex hull of a finite set of points in the plane, Information Processing Letters 2 (1973) 18–21. February 21, 2013.
[7] F. P. Preparata, S. J. Hong, Convex hulls of finite point sets in two and three dimensions, Communications of the ACM 2 (20) (1977) 87–93.Febraury 24, 2013.
[8] R. Wenger, Randomized quick hull, Algorithmica 17 (1997) 322–329. March 02, 2013.
[9] Herv´e Br¨onnimann, John Iacono, Jyrki Katajainen, Pat Morin, Jason Morrison and Godfried Toussaint. Space-Efficient Planar Convex Hull Algorithms (2007-08). March 12, 2013.
[10] D. G. Kirkpatrick and R. Seidel. The ultimate planar con vex hull algorithm.SIAM J. Comput., 15(1):287–299, 1986. March 16, 2013.
[11] T. M. Chan. Dynamic planar convex hull operations in near-logarithmic amortized time. Journal of the ACM, 48(1):1–12. May 02, 2013.
[12] Gerth Stølting Brodal, Riko Jacob. Dynamic Planar Convex Hull. May 21, 2013.
[13] K. H., R. Tarjan, and T. K. Faster kinetic heaps and their use in broadcast scheduling. In Proc. 12th ACM-SIAM Sympo-sium on Discrete Algorithms, pages 836–844, 2001. April 13, 2013.
[14] A. M. Andrew. Another efficient algorithm for convex hulls in two dimensions. Information Processing Letters,9(5):216–219, 1979. April 24 2013.
[15] C. Bajaj and M.-S. Kim. Convex Hulls of Objects Bounded by Algebraic Curves Algorithmica,Vol. 6, pp. 533{553, 1991. May 18, 2013.
[16] M. do Carmo. Dierential Geometry of Curves and Surfaces.Prentice-Hall, 1976. April 19, 2013.
[17] D. Dobkin and D. Souvaine. Computational Geometry in a Curved World.Algorith-mica,Vol. 5, No. 3, pp. 421{457, 1990. June 01, 2013.
[18] D.T. Lee. On Finding the Convex Hull of a Simple Polygon. Int'l J. Computer and Information Sciences,Vol. 12, No. 2, pp. 87{98, 1983. May 29, 2013.
[19] E. Sherbrooke and N. Patrikalakis. Computation of The Solutions of Nonlinear Polyno-mial Systems.Computer AidedGeometric Design,Vol 10, No 5, pp 379. April 27, 2013.
[20] M. H. Overmars and J. van Leeuwen. Maintenance of con-figurations in the plane. J. Comput. System Sci., 23(2):166–204, 1981. December 30, 2012.
[21] Yao, A.C. (1981). A lower bound to finding convex hulls. Journal of the ACM 28, 780-789. February 18, 2013
[22] Bentley, J.L. and M.I. Shamos (1978). Divide and conquer for linear expected time. Information Processing Letters 7, 87-91. March 10, 2013
[23] http://en.wikipedia.org/convexhull, April 18, 2013
[24] http://www.personal.kent.edu/~rmuhamma/Compgeometry, April 20, 2013
[25] http://en.wikipedia.org/grahamscan, May 11, 2013
[26] http://www.topcoder.com/tc, June 06, 2013 Control (ICICIC2008), June 2008.
[27] A sublogarithmic convex hull algorithm. Journal of the Springer, BIT Numerical Mathematics Volume 30, Issue 3, pp 378–384. September 1990.
[28] https://www.slideshare.net/kasunrangawijeweera/an-efficient-convex-hull-algorith, June 3, 2014.
[29] Gang Mei, John C.Tipper, Nengxiong Xu, An Algorithm for Finding Convex Hulls of Planar Point Sets. Proceedings of IEEE Conference, pp.888{891, 29-31 Dec. 2012.